Finite State Intensional Semantics

Mats Rooth
Cornell University
mr249@cornell.edu

1 Introduction

Suppose possible worlds are strings, rather than physically structured worlds like ours. Then the proposition corresponding to a sentence or a formula in logical language is a set of strings; an epistemic acquaintance relation is a relation between strings; and in a relational construction of partition semantics for questions, a question meaning is a relation between strings. If discourse referents, too, are encoded in the same strings, then dynamic information states are sets of strings, and distributive updates are relations between strings. Finally, in a construction of the temporal development of worlds in an action logic, actions are also relations between strings.

This paper makes the further assumption that the sets and relations mentioned above are regular sets and regular relations. These are the sets of strings and relations between strings that can be represented by finite state acceptors and transducers. In this framework an analysis of intensional complementation in *that-* and *wh-* clauses is developed. A version of dynamic semantics is used to formalize sub-clausal compositional semantics. The analysis is cast as a defined logical language in a finite state calculus that includes function definitions. Finally, an English fragment that maps to the logical language is formulated in an extended categorial grammar.

In a running example, a robot walks through a string of characters with a sequence of turns and steps, and picks up information by looking at the character in front of it. A world state is a string such as `>+b._-a_-t_`, where the letter string is `bat` and the character > represents the robot facing right at the start of the string. We conceive of this world as resulting from the robot starting in the configuration `<-b._-a_-t_`, then turning and looking at the letter in front. In this world, the extensional sentence (1a) is true. The intensional sentence (1b) is false, because world `>+b._-a_-b_` is an epistemic alternative for the robot, and in that word the complement sentence (2c) is false.

(1) a. A B precedes an A that is adjacent to a T.
    b. He knows that there is a T in position three.
    c. There is a T in position three.

A semantics for English sentences will be obtained by parsing sentences with a categorial grammar, to assign the logical forms (2a,b) for (1a,b). Then in the finite state calculus, a set of strings as represented by a finite state machine is computed for each LF. This is a model-theoretic propositional content for the sentence or formula. Finally, truth values in particular worlds are found by checking membership in the propositions.

(2) a. $\mathit{Indef}(\mathit{Let}(t), \mathit{Indef}(\mathit{Intersect}(\mathit{Let}(a), \mathit{Bind}(\mathit{Trace}(\mathit{Indef}(\mathit{Let}(b), \mathit{Pre}))))), \mathit{Adj}))$
    b. $\mathit{Kt}(\mathit{Indef}(\mathit{Let}(a), \mathit{Indef}(\mathit{Pos}(3), \mathit{In})))$

The paper is organized like this. Section 2 presents a statement of Hintikka semantics for intensional complements in the finite state framework. Section 3 looks at sub-clausal compositional semantics, using an encoding of discourse referents in strings. Section 4 presents the syntax-semantics interface, based on a categorial grammar. Section 5 extends the semantics to question complements, using partition semantics for questions. Section 6 describes the implementation. Section 7 comments on applications and related work.

2 Intensional complementation

A regular Kripke frame is a tuple $\langle \Sigma, W, R \rangle$, where Σ is a finite alphabet, W is a regular subset of Σ^*, and R is a regular relation on W. This is an ordinary Kripke frame, where the set of worlds W is a regular set of strings, and the accessibility relation R is a regular relation. Suppose we are given the semantic value of the complement clause ϕ in (3a) as a regular subset of W, and want to define the semantic value of the whole sentence as a set of strings. Or for the LF language, we are given the semantic value (3b) and want to define (3c).

(3) a. He knows that ϕ.
    b. $[\![\phi]\!]$, a regular subset of W
    c. $[\![K(\phi)]\!]$, a regular subset of W

Hintikka semantics for intensional complementation holds $K(\phi)$ to be true in w if and only if ϕ is true in every world v that is an alternative in w. This is stated using world quantification in (4). We want an equivalent semantics in the finite state calculus, which is an algebra of regular sets of strings and regular relations on strings, with operations including Boolean operations, composition of relations, Kleene closure, domain and co-domain of relations, and restriction of relations to domains and co-domains.[1] (5) defines $K(\phi)$ using set difference ($X - Y$), domain of a relation ($Do(R)$), and restriction of a relation R to a codomain X ($R \circ X$). It subtracts from W those worlds that are R-related to a world where ϕ is false. The latter is enforced by restricting R to the codomain $W - [\![\phi]\!]$.

(4) $$[\![K(\phi)]\!] = \{w | \forall v. R(w, v) \rightarrow [\![\phi]\!](v)\}$$

(5) $$[\![K(\phi)]\!] = W - Do(R \circ (W - [\![\phi]\!]))$$

In the running example, R is a knowledge modality, so it should be an equivalence relation. Beyond this, it should tie in a particular way to the development of world states. As the robot acts, it never loses information. Second, it picks up information by knowing how it acts, and by looking at the letter in front of it. This provides for a way of encoding the epistemic state of the robot in the world string. To capture the epistemic state of the robot, all that matters is where the robot is, how it is oriented, what letters the robot has looked at. In a world state like `>+b._-a_-t_`, plus characters + precede the letters that the robot has looked at, and minus characters − precede the characters that it has not looked at. We assume the initial state for any world-time line has only minus marks, indicating that the robot has no a-priori knowledge about the letter sequence in its world.

To develop the running example, define the set of worlds W by the sequence of definitions in (6). A world is a character sequence with CVC shape, where C is either `t` or `b`, and V is either `a` or `e`. Each letter is preceded by a sign + or −, indicating whether the robot has looked at that letter. Finally each plus/minus sign is preceded by a slot for the agent. This is a character position containing either < (robot facing left), > (robot facing right), or _ (no robot). A character position of this same shape also finishes the string. The conjunct $[\cdot * A_1 \cdot *]$ in the definition of W at the bottom in (6) ensures that a world string contains at least one robot. The center dot is a variable over all characters, and the star is asteration (Kleene star). The subtracted term $[\cdot * A_1 \cdot * A_1 \cdot *]$ ensures that a world string does not contain two robots. The syntax maps directly to the Fst language that is implemented in the Xfst and Foma interpreter-compilers (Beesley and Karttunen, 2003; Hulden, 2009). See the code and replication material Rooth (2017a,b).

[1] We assume the finite state calculus that is implemented in Xfst and Foma (Beesley and Karttunen, 2003; Hulden, 2009). Function definitions in Xfst are described in Karttunen (2010). For definition (5), an algebra is needed with Boolean set operations, domain for relations, and restriction of a relation to a co-domain. Later parts of the paper use relation composition.

(6) $C :=$ t | b
$V :=$ a | e
$S :=$ + | -
$A_1 :=$ < | >
$A_2 :=$ < | > | _
$W :=$ $[[A_2SC][A_2SV][A_2SC]A_2] \wedge [\cdot* A_1 \cdot*] - [\cdot* A_1 \cdot* A_1 \cdot*]$

R in the running example is defined by making substitutions for letters that are marked by a minus sign, i.e. letters in positions that the robot has not seen. Letters that the robot has seen remain constant when moving to an epistemic alternative. In (7), substitution is stated using the rewrite notation of the finite state calculus, which is a defined notation for relations that make substitutions in specified contexts (Kaplan and Kay, 1994; Kempe and Karttunen, 1996). Rewriting in the finite state calculus is inspired by the rewrite notation of classical generative phonology (Chomsky and Halle, 1968).

(7) $R_c := C \rightarrow C \parallel -\_\_$
$R_v := V \rightarrow V \parallel -\_\_$
$R := W \circ R_c \circ R_v \circ W$

That R is an equivalence relation can be shown by reasoning about the definition, and can also be checked computationally for equal-length relations. (8) defines properties characterizing a relation as being reflexive, symmetric, and transitive, with equivalence relations satisfying all of these. Such properties in the finite state calculus are modeled as functions that map to truth values, where *False* is the empty set and *True* is the unit set of the empty string. The definitions use a conditional in the finite state calculus, which is defined in (9). R being an equivalence relation is checked by evaluating $Eqv(R)$.

(8) $Rfl(U) := If(Id_W - (U \wedge Id_W), False, True)$
$Sym(U) := If(U - Conv(U), False, True)$
$Trs(U) := If((U \circ U) - U, False, True)$
$Eqv(U) := If(Rfl(U) \wedge Sym(U) \wedge Eqv(U), True, False)$

(9) $If(X, Y, Z) := (Z - Co(X \circ (\cdot* \times \cdot*))) \mid (Y \wedge Co(X \circ (\cdot* \times \cdot*)))$

The above definition of R leaves the robot in the same place and orientation, and keeps the length of the world constant. So in this example, the robot has a priori knowledge of its location and of the size of the world. Nothing about this is essential: the general account assumes nothing about W and R, beyond being a regular set and a regular relation on it. This is as usual in Kripke semantics. It is possible for W to be countably infinite—the replication supplement includes a version of the running example where the size of worlds in not bounded.

3 Compositional semantics

Type theoretic approaches to compositional semantics use functions to express lexical content and the semantics of complex phrases (Montague, 1973; Janssen, 1996). While function definitions are used in the finite state calculus in the way illustrated above, functions are not first class objects that themselves can be the arguments of functions or algebraic operators—really there are only one or two types in the finite state calculus. Further, the modal space of world-strings as presented above does not represent individuals. What then is the interpretation of the constants *Adj* (for adjacent) and *Pre* (for precede)? The approach taken here is to include in the model specification a set W_1, which we think of as a space of worlds with a distinguished individual. Similarly for a set W_2, a space of worlds with two distinguished individuals. In the running example, an element of W_1 is a string with an index 1 inserted after some letter, interpreted as a discourse referent for that token letter. So <+b._-a1_-b_ is the world <+b._-a_-b_ with a discourse referent for the middle letter. Using terminology from dynamic semantics, in the extended world <+b._-a1-b_, the token letter a is the center (e.g. Bittner 2003). <+b._-a1_-b2_ is the world <+b._-a_-b_ with a center for the second letter, and a secondary center

for the third letter. In the running example, *Fol* is a subset of W_2 that is used as an interpretation for *follow*. The object is encoded as the center, and the subject as the secondary center (or pericenter). *Fol* includes the two-centered worlds listed in (10).

(10)
```
_+b1_-a2>-t._  _-t1<-e2_+b._  _-b._-a1_-b2>  _-t._-a1_+t2>
_-t1_-e2>-t._  _-t1<-e2_+b._  _+t._+a1>+t2_  _-b1_-e2>-b._
_-t1_-e2<+t._  _-b._-a1_-t2>  _-b._+e1_+b2<  _+b1>+a2_+b._
_+b1<+a2_+b._  _+t._+e1<+t2_  _+t._-e1>+t2_  _-t1_-a2>-t._
```

The function *Indef*(X, Y) seen in (11) is used as a co-predicator on the center. It intersects its arguments, and then pops the center, with the input pericenter if present being the output center. (11a) is a subset of W that includes the worlds listed in (11b). *Indef* is used as a predicator both for the object and the subject.

(11) a. *Indef*(*Let*(t), *Indef*(*Let*(a), *Fol*))

b.
```
>+b._+a._+t._  _-t.<+a._+t._  <-t._-a._-t._
>+t._-a._-t._  _+t._-a.<+t._  <+b._+a._-t._
_+b.<-a._-t._  _+b._-a.<-t._  >+b._+a._+t._
```

Traces are treated along the same lines, using the marker 0 for traces. The operator *Trace* maps the center to the trace center, and maps the pericenter (if present) to the center. *Trace*(*Fol*) conceptually has a trace in the object position of *follow*, and *Trace*(*Indef*(*Let*(a), *Fol*)) conceptually has a trace in the subject position of *follow*. (12) and (13) show samples of these propositions.

(12) a. *Trace*(*Fol*)

b.
```
_-b0_+a1_+b.<  _+t._+e0_+t1>
<+t0_-e1_-t._  _-b._+e0<-b1_
```

(13) a. *Trace*(*Indef*(*Let*(a), *Fol*))

b.
```
_+b._-a.<+t0_  _-t._+a.<-t0_
<-b._-a._+b0_  _-t._-a._-t0>
```

The *Bind* operator seen in (2a) maps the trace center to the ordinary center, converting a phrase with a free trace to a property of individuals. The *Intersect* operator intersects two properties of individuals, and is used to combine a noun with a relative clause.

See the end of Section 6 for a complex LF that includes most of the operators.

4 Syntax-semantics interface

The material from Section 2 and Section 3 is concerned with a formal language with terms such as $K($*Indef*(*Let*(a), *Indef*(*Let*(t), *Fol*$)))$. The terms get a model-theoretic interpretation as regular sets of strings. To tie this in with natural language, lexical items, phrases, and sentences of the natural language should be mapped to the formal language. Here this is accomplished with a categorial grammar. (15) lists some simple lexical entries, with word forms in in the first column, categorial types in the second column, and logical terms in the third. $e\backslash_N t$ is the categorial type for a noun. It is a multimodal slash type that is not active as a function in the grammar—it enters into a derivation only as an argument of a determiner. In type theoretic semantics, terms of syntactic category $e\backslash_N t$ have semantic type et, the type of functions from individuals to truth values. We keep the syntactic category, but interpret phrases of that category as subsets of W_1. $(e\backslash t)/e$ is the categorial type label for a transitive verb. In type theoretic semantics, this corresponds to type eet, while in the present system, it is semantically a subset of W_2.[2]

[2]The type labels et and eet are written in the notation from Link (1979), with right association and without commas or brackets.

(14) A $e\backslash_N t$ *Let*(a)
B $e\backslash_N t$ *Let*(b)
follow $(e\backslash t)/e$ *Fol*

The indefinite determiner is in the lexicon twice, once for use with an object, and once for use with a subject. In each case the semantics uses *Indef*, the co-predication operator on the center. See (15).

(15) a $(((e\backslash t)/e)\backslash(e\backslash t))/(e\backslash_N t)$ $\lambda X\lambda Y.$*Indef*(X,Y) object determiner
a $(t/(e\backslash t))/(e\backslash_N t)$ $\lambda X\lambda Y.$*Indef*(X,Y) subject determiner

(16) is the first part of the analysis of traces and binding. The syntactic category of a clause with a bound trace is $e\backslash_T t$. This is again a nominally functional slash type that is not active as a function in the grammar. The relative morpheme *that* combines with $e\backslash_T t$ on the right, and a noun $e\backslash_N t$ on the left, to form a noun $e\backslash_N t$.

(16) that $((e\backslash_N t)\backslash(e\backslash_N t))/(e\backslash_T t)$ $\lambda X\lambda Y.$*Intersect*(X,Y)

(17b) is the target LF for the relative clause (17a), with an object trace. The object trace should trigger the operator in the term *Trace*(*Fol*), converting the center to a trace center. At the level of the relative clause, the *Bind* operator should convert the trace center back into an ordinary center. Thus the trace has both a local semantic effect, and an effect at its scope level. This is enforced in Barker and Shan's continuation scope calculus, using the categorial type $(e\backslash_T t)/\!\!/(((e\backslash t)/e)\backslash(e\backslash t)\backslash\!\!\backslash t)$. This indicates a local type $((e\backslash t)/e)\backslash(e\backslash t)$, which combines with a transitive verb (type $(e\backslash t)/e$) to form a predicate (type $e\backslash t$). Second there is a scope shell $(e\backslash_T t)/\!\!/(-\backslash\!\!\backslash t)$. This indicates that the trace takes scope at the clausal type t to form the bound-trace category $e\backslash_T t$. The first line in (18) is the lexical entry for the object trace. The semantics $\lambda k.$*Bind*$(k($*Trace*$))$ has the effect of applying *Trace* to the verb meaning *Fol* downstairs, because the continuation variable k is applied to *Trace* in the term $k($*Trace*$)$. The second line in (18) is the analogous lexical entry for a subject trace. The traces are in the lexicon with the spelling "e".

(17) a. a B follows e $(e\backslash_T t)$
b. *Bind*(*Indef*(*Let*(b), *Trace*(*Fol*)))

(18) e $e\backslash_T t/\!\!/(((e\backslash t)/e)\backslash(e\backslash t)\backslash\!\!\backslash t)$ $\lambda k.$*Bind*$(k($*Trace*$))$ object trace
e $e\backslash_T t/\!\!/((t/(e\backslash t))\backslash\!\!\backslash t)$ $\lambda k.$*Bind*$(k($*Trace*$))$ subject trace

(19) a. e follows a B, $(e\backslash_T t)$
b. *Bind*(*Trace*(*Indef*(*Let*(b), *Fol*)))

Some observations are in order about the system of categories and the semantics. Familiar syntactic categorial symbols are used, such as $(e\backslash t)/e$ for a transitive verb. However, the corresponding semantics is not ultimately used as a function of type eet, contrary to what happens in type-theoretic interpretation using function application. The reason is that a term of the form *Fol*(x) is not a term of the target logical language. The semantic terms in (15), (16), and (18) use lambda. These terms are meaningful in an extension using variables and binding by lambda of the logical language described in Section 2 and Section 3. However, since it is the basic logical language that is targeted, it is important that the lambdas are eliminated by beta reduction in complete derivations.

5 Question complements

In partition semantics for questions, a question complement contributes an equivalence relation on worlds (Groenendijk and Stokhof, 1984). Suppose ϕ is a question complement associated with the relation Q. Informally, w and w' are related by Q if and only if the complete answer to the question is the same in w and w'. Here we will use regular relations between worlds as question denotations.

In a whether-question such as the complement in (20a), there are two cells in the partition associated with the equivalence relation, one consisting of worlds where the clause (20b) is true, and the other consisting of worlds where the clause is false. The corresponding relation is defined in the finite state calculus using Cartesian product and union, see (21).

(20) a. He knows whether there is a B in three.
b. There is a B in three.

(21) $\mathit{Whether}(X) := \quad (X \times X) \mid ((W - X) \times (W - X))$

The other half of the problem is to define a semantics for question-embedding *know* in terms of the epistemic accessibility relation R, and a relation between worlds contributed by a wh-complement. The robot knows whether Q if his knowledge as represented by R resolves the question encoded by Q. This is true in a given world w when the set of worlds that are R-related to w are all within a single cell of the partition corresponding to Q. Let $\bar{Q}$ be the complement of Q, which relates worlds that are in different cells of the partition. Let Id_W be the identity relation on worlds. The relation $R \circ \bar{Q} \circ R^{-1}$ relates pairs (w, w') such that one can start at w, jump to an epistemic alternative to w in some cell of the partition, jump using $\bar{Q}$ to a different cell of the partition, then jump back to w' using R^{-1}, the inverse of R. Intersecting with Id_w gives us the set of pairs (w, w) such that w is related by R to worlds in different cells of the question partition. This characterizes worlds w where the robot does not know whether Q. The set of worlds where the robot knows whether Q is the complement. This leads to the definition (23) for question-embedding *know*.[3]

(22) $\mathit{Kw}(Q) := \quad W - \mathit{Do}((R \circ (W \times W - Q) \circ R^{-1}) \wedge \mathit{Id}_W)$

Unfortunately, the relative complement $(Q' - Q)$ is not defined in the finite state calculus for arbitrary relations Q' and Q, only for equal-length relations. These are relations where any pair of related strings have the same length. The running example has worlds with fixed length, so all relations between worlds are length-preserving. But since we want to include the possibility of countably infinite sets of worlds, it is not desirable to rely on worlds having a fixed string length.[4] We are better off if the question complement contributes the complement relation to begin with. This is easy enough for whether-complements. See the revised definitions (23) and (24).

(23) $\mathit{Whether}_2(X) \quad := \quad (X \times (W - X)) \mid ((W - X) \times X)$

(24) $\mathit{Kw}_2(Q) \quad := \quad W - \mathit{Do}((R \circ Q \circ R^{-1}) \wedge \mathit{Id}_W)$

We would also like to interpret embedded constituent questions such as the ones in (25). This is not in the current system.

(25) a. He knows what letter is in position three.
b. He knows what vowels follow what consonants.

[3] If R is assumed to be symmetric, R can be substituted for R^{-1}. In the finite state calculus as implemented in Xfst and Foma, the identity relation on a given set is not distinguished from the set, so W can be substituted for Id_W. For the same reason, the domain operator *Do* can be dropped.

[4] Possibly the problem can be partially finessed, by including for a world w of basic length n also a world of lenght $n + m$ that is obtained from w by adding m dummy characters.

6 Computational implementation

The proposal is implemented using a parser for categorial grammar, and a toolkit for the finite state calculus. The parser is Barker and Shan's parser for continuation categorial grammar (Barker and Shan, 2005). The grammar is a lexicon given in Scheme-Lisp format, consisting of a list of tuples of word forms, syntactic category symbols, and semantic terms. (26) is the lexical entry that corresponds to the first version of the indefinite article in (15). This lexical entry and the rest are isomorphic to what was seen in Section 4.

(26)
```
(("a" ((((e \ t) / e) \ (e \ t)) / (e \N t))) (^ P (^ f (Indef P f))))
```

A sentence is presented to the parser with a Scheme parse command (see (27b). If the parser finds a derivation, an LF is printed in Lisp format, see (27c). This is converted to the logical language using a Lex program, by moving parentheses and inserting commas, to obtain (27d). Now this formula is read as a term denoting a regular set into an interpreter for the finite state calculus, in an environment where functions have been defined in the way characterized in Sections 2 and 3. Xfst or Foma is used as the interpreter for the finite state calculus. To illustrate the proposition that results, a random set of elements of the proposition are printed. Or one can test entailment or equivalence between propositions contributed by two sentences. (28) shows the procedure for printing a sample of the proposition in Xfst. (In the code the letter nouns are written "ay" and "bee", rather than "A" and "B".)

(27) a. a ay that a bee follow e follow a tee

b.
```
(parse '(a ay that a bee follow e follow a tee))
```

c.
```
(Indef (Intersect (Bind (Indef (Let b) (Trace Fol))) (Let a))
  (Indef (Let t) Fol))
```

d. $\mathit{Indef}(\mathit{Intersect}(\mathit{Bind}(\mathit{Indef}(\mathit{Let}(b),\mathit{Trace}(\mathit{Fol}))),\mathit{Let}(a)),\mathit{Indef}(\mathit{Let}(t),\mathit{Fol}))$

(28)
```
regex Pr(Indef(Intersect(Bind(Indef(Let(b),Trace(Fol))),Let(a)),
                Indef(Let(t),Fol)));
xfst[17]: print random-words
_+t._-a._-b.<
_-t._+a.>+b._
_+t._-a.>-b._
_-t._-a._-b.>
_-t._+a.<+b._
...
```

These steps are tied together using file interfaces. With the input sentence in an input file `s1.snt`, parsing can be triggered from a make file, with the LF written into a file `s1.lf`, and the sample written into a file `s1.wld`. See the examples in the replication supplement.

In the implementation of the running example, a term is actually not interpreted directly as a set of strings like the ones seen at the end in (28), and elsewhere in the paper. Instead, a world is a 15-element bit vector that can be printed to a world string of the other kind, see (29.)[5] This explains the presence of the operator `Pr` in (28). The motivation for this is two-fold. First, it results in more structured definition of the modal space, based on the primitive propositions listed in (30).

(29)
```
regex Pr({011110110000000});
xfst[19]: print words
>+t._+a._-b._
```

[5] Curly brackets in {`011110110000000`} explode the contained string into the character sequence `0 1 1 1 1 0 1 1 0 0 0 0 0 0 0`.

(30)
A	b vs. t is in letter position 1
B	a vs. e is in letter position 2
C	b vs. t is in letter position 3
D	letter position 1 has been seen
E	letter position 2 has been seen
F	letter position 3 has been seen
G	robot is in position 1 or 2 (and not 3 or 4)
H	robot is in position 1 or 3 (and not 2 or 4)
I	robot is facing left
J	dref2 in 2 or 3 (vs dref2 in 1 or no dref2)
K	dref2 in 1 or 3 (vs dref2 in 2 or no dref2)
L	dref1 in 2 or 3 (vs dref1 in 1 or no dref1)
M	dref1 in 1 or 3 (vs dref1 in 2 or no dref1)
N	trace dref in 2 or 3 (vs trace dref in 1 or no trace dref)
O	trace dref in 1 or 3 (vs trace dref in 2 or no trace dref)

A second motivation has to do with a goal of incorporating actions and the temporal development of worlds in the modal space. The idea is to assume four primitive actions Step (the robot stepping forward), Turn (the robot turning), Look (the robot picking up information from the position in front of it) and R. The latter is a pseudo-action of non-deterministically moving to an epistemic alternative. In a development of the action logic in Kleene algebra with tests (Kozen, 1997), the Kleene elements are *Step*, *Turn*, *Look*, and the propositions in (30) are generators for the Boolean algebra of tests. A world at a time is a sequence of Kleene elements (actions), with interleaved stative propositions. Models with this structure are generally comparable to the action and epistemic models that figure in research on situation calculus (Moore, 1984; Reiter, 2001; Scherl and Levesque, 2003; Levesque and Lakemeyer, 2008). They have the specific structure of trace models for Kleene algebra with tests. We have developed a partial axiomatization of the logic based on an implementation of KAT with hypotheses (Pous, 2015). This is strictly work in progress, but is relevant here as motivation for the definition of the finite state semantics using generating propositions.

Figure 1 and (31) give an abbreviated derivation for the complex sentence (31a). In the figure, only phrases that have an interpretation in the finite state calculus are listed. A sample is included of four words from the proposition that corresponds to the LF in the example model. The proposition corresponding to (31a) and (31b) is a set of eight worlds, which are listed in (31c). They differ just in the position and orientation of the robot, giving 4×2 worlds. Each of them has the letter sequence `tab`, and in each of them the robot has seen each letter. Moreover, restricted to this set, R is the identity relation. In other words, in the example model, (31b) is true only when the robot has identified its world.

(31) a. He know that a tee adjacent a ay that follow a bee.

b. $K(\mathit{Indef}(\mathit{Let}(t), \mathit{Indef}(\mathit{Intersect}(\mathit{Bind}(\mathit{Trace}(\mathit{Indef}(\mathit{Let}(b), \mathit{Fol}))), \mathit{Let}(a)), \mathit{Adj})))$

c.
```
<+t._+a._+b._
>+t._+a._+b._
_+t._+a.<+b._
_+t._+a.>+b._
_+t._+a._+b.<
_+t._+a._+b.>
_+t.>+a._+b._
_+t.<+a._+b._
```

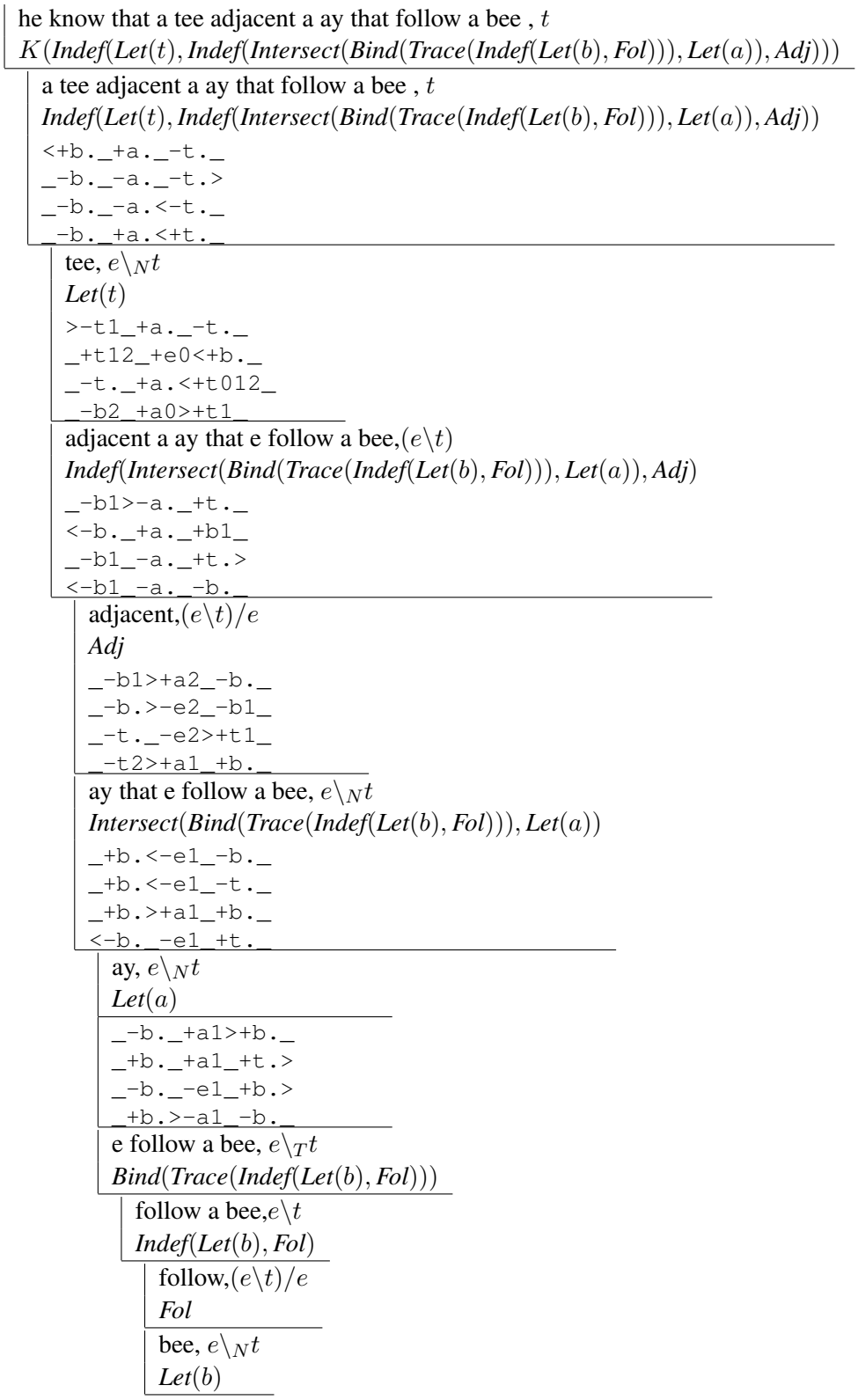

Figure 1: Abbreviated derivation for a complex sentence, including word sequence, syntactic category, LF, and a sample of four words from the proposition.

7 Discussion

The material presented here has been used in a second semester graduate course in linguistic semantics. The course adds computational topics and methodology to the curriculum on compositional semantics and possible worlds semantics. This is done in part by using parsers and derivation calculators, including a categorial-grammar parser (Barker and Shan, 2005), and a derivation calculator that interprets logical forms in a typed intensional logic (Champollion et al., 2013). Finite state intensional semantics adds the possibility of actually computing propositional denotations as sets of worlds, rather than just computing logical terms associated with phrases. In addition to the material on intensional complementation and questions, the course included computing with premise semantics for modality (Kratzer, 1981; Lewis, 1981), in a framework where the ordering propositions are a finite set of regular propositions.

Research and publications in linguistic semantics frequently refer to toy possible worlds models with a handful of worlds, in order to illustrate and test ideas. The methods introduced here allow this to be scaled up, even to countable sets of worlds, and to avoid error, by virtue of the formal methodology. And they start to bridge ideas about model structures and computation between possible worlds semantics as applied in natural language semantics, and the research program of cognitive robotics, where constructed possible worlds models are used to reason about action, planning, modality, and knowledge (Moore, 1984; Reiter, 2001; Scherl and Levesque, 2003; Levesque and Lakemeyer, 2008).

Tim Fernando and Lauri Carlson have developed approaches to natural language semantics using finite state methods (Fernando, 2007; Carlson, 2009). They mostly focus on tense, aspect, and the structure of events, rather than as here the syntax-semantics interface for intensional complementation. Fernando discusses intensions in his system of finite state semantics in Fernando (2017). Comparing and integrating the approaches is an important topic for future work.

The epistemic model in the running example is constructed to reflect certain intuitions, but this is done in an ad-hoc way. The topic of constructing epistemic action models in a systematic way was approached by Baltag et al. (1999) using action alternatives, and a large body of research has followed. See Van Ditmarsch et al. (2007) and Van Ditmarsch et al. (2015) for surveys. It is hoped that connecting with this literature and methodology will lead to richer examples, and an expansion in the set of embedding verbs that are can be covered.

Systems of quantified modal logic and intensional typed interpretation use models built from a set of worlds and a set of individuals (Lewis, 1968; Gallin, 2011). Where are the individuals in the current system? In Section 3, the strings in W_1 were described as worlds with a distinguished individual, or worlds with a discourse referent marking an individual. It is possible to directly view W_1 as the set of individuals. In model structures as defined by Lewis, individuals are not shared between possible worlds, and each individual can be mapped to its world. This suggests using model frames $\langle \Sigma_w, \Sigma_D, W, D, \pi, R, C \rangle$, where Σ_w and Σ_D are alphabets, W is a regular subset of Σ_W^*, D is a regular subset of Σ_D^*, π is a functional regular relation between D and W, and R is a regular relation on W, and C is a regular relation on Σ_D. π maps individuals to their worlds. C is the cross-world counterpart relation. It is hoped that the addition of C will make it possible to include de re LFs with *know* in the analysis. Another part of the analysis of *know* is presupposition. Collard (2016) introduced a version of the finite state system explained here that uses three-valued evaluations, in order to model presupposition.

Acknowledgments

Thanks to Jacob Collard, Todd Snider, Ede Zimmermann, and three IWCS 2017 reviewers for comments. Prior to the conference, this matierial was presented at the Goethe University/Frankfurt in July, 2017. Thanks to the audience for their comments and reactions.

References

Baltag, A., L. S. Moss, and S. Solecki (1999). The logic of public announcements, common knowledge, and private suspicions.

Barker, C. and C.-c. Shan (2005). Reference parser for explaining crossover and superiority as left-to-right evaluation. URL `semanticsarchive.net/Archive/TI1M2UxN`.

Beesley, K. R. and L. Karttunen (2003). *Finite State Morphology*. Center for the Study of Language and Inf.

Bittner, M. (2003). Word order and incremental update. In *Proceedings from the Annual Meeting of the Chicago Linguistic Society*, Volume 39, pp. 634–664. Chicago Linguistic Society.

Carlson, L. (2009). *Tense, Mood, Aspect, Diathesis*. Book ms., University of Helsinki.

Champollion, L., J. Tauberer, M. Romero, and D. Bumford (2013). The lambda calculator for students and teachers of natural language semantics. URL `github.com/nyusemantics/LambdaCalculator`.

Chomsky, N. and M. Halle (1968). *The Sound Pattern of English*. ERIC.

Collard, J. (2016). A finite state model of semantics with presupposition. Manuscript, Cornell University.

Fernando, T. (2007). Observing events and situations in time. *Linguistics and Philosophy 30*(5), 527–550.

Fernando, T. (2017). Intensions, types and finite-state truthmaking. In *Modern Perspectives in Type-Theoretical Semantics*, pp. 223–243. Springer.

Gallin, D. (2011). *Intensional and Higher-order Modal logic: With applications to Montague semantics*, Volume 19. Elsevier.

Groenendijk, J. and M. Stokhof (1984). *On the Semantics of Questions and the Pragmatics of Answers*. Foris.

Hulden, M. (2009). Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pp. 29–32. Association for Computational Linguistics.

Janssen, T. (1996). *Compositionality*. Institute for Logic, Language and Computation (ILLC), University of Amsterdam.

Kaplan, R. M. and M. Kay (1994). Regular models of phonological rule systems. *Computational linguistics 20*(3), 331–378.

Karttunen, L. (2010). Update on finite state morphology tools. *Ms., Palo Alto Research Center*.

Kempe, A. and L. Karttunen (1996). Parallel replacement in finite state calculus. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pp. 622–627. Association for Computational Linguistics.

Kozen, D. (1997). Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems (TOPLAS) 19*(3), 427–443.

Kratzer, A. (1981). The notional category of modality. *Words, worlds, and contexts*, 38–74.

Levesque, H. and G. Lakemeyer (2008). Cognitive robotics. *Foundations of artificial intelligence 3*, 869–886.

Lewis, D. (1981). Ordering semantics and premise semantics for counterfactuals. *Journal of philosophical logic 10*(2), 217–234.

Lewis, D. K. (1968). Counterpart theory and quantified modal logic. *the Journal of Philosophy 65*(5), 113–126.

Link, G. (1979). *Montague-Grammatik Die Logische Grundlagen*.

Montague, R. (1973). The proper treatment of quantification in ordinary english. In *Approaches to Natural Language*, pp. 221–242. Springer.

Moore, R. C. (1984). A formal theory of knowledge and action. Technical report, DTIC Document.

Pous, D. (2015). Symbolic algorithms for language equivalence and kleene algebra with tests. In *ACM SIGPLAN Notices*, Volume 50, pp. 357–368. ACM.

Reiter, R. (2001). *Knowledge in Action: Logical foundations for specifying and implementing dynamical systems*. MIT press.

Rooth, M. (2017a). Finite-state-intensionality. URL `github.com/MatsRooth/Finite-state-intensionality`.

Rooth, M. (2017b). Replication data for: Finite state intensional semantics. Harvard Dataverse Network. DOI `10.7910/DVN/EV6KLF` URL `http://dx.doi.org/10.7910/DVN/EV6KLF`.

Scherl, R. B. and H. J. Levesque (2003). Knowledge, action, and the frame problem. *Artificial Intelligence 144*(1-2), 1–39.

Van Ditmarsch, H., J. Y. Halpern, W. van der Hoek, and B. P. Kooi (2015). *Handbook of Epistemic Logic*. College Publications.

Van Ditmarsch, H., W. van Der Hoek, and B. Kooi (2007). *Dynamic Epistemic Logic*, Volume 337. Springer Science & Business Media.